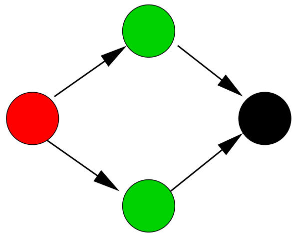
NSA's HCSS, Maritime Institute May 2007
based on MILS Integration PP, presentation to OG RTES, 1 February 2007, San Diego based on presentation to OG RTES, 26 October 2006, Lisbon.

Compositional Certification for MILS

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

Intuitive Security Architecture



- Almost all system designs are portrayed in diagrams using circles and arrows
- But in security, these have a particular (often unconscious) force and interpretation
- Arrows indicate interfaces
  - Implicitly, absence of an arrow means absence of component interaction
- Circles indicate encapsulated data, information, control, etc.
  - The only things that happen inside a circle are consequences of things in that circle and the incoming arrows, and the only things that change are the internal state of the circle and its outgoing arrows

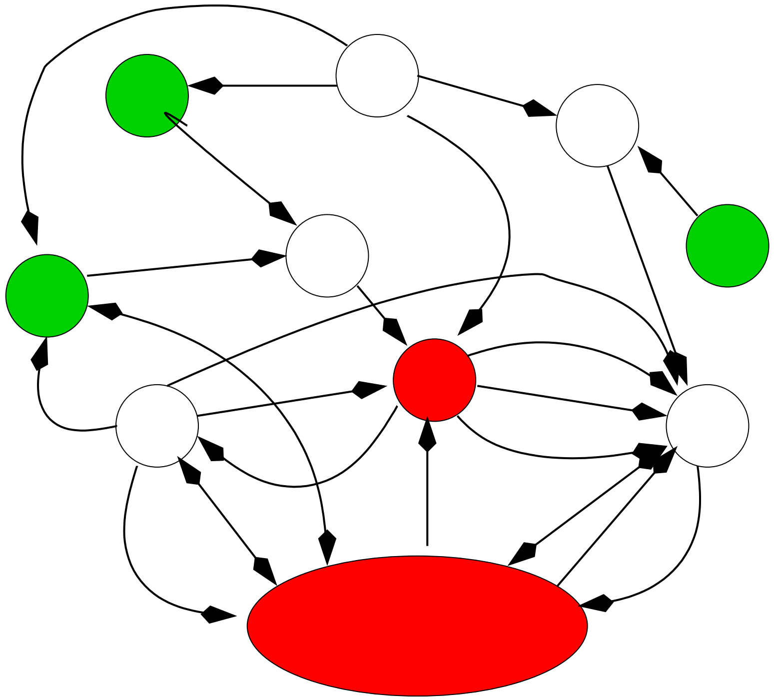
Good Intuitive Security Architecture

- Try to arrange the circles and arrows so that

  security depends on only a few trusted circles

- And those are trusted to do only relatively simple things

- Split big circles up if necessary to achieve these



The MILS Idea

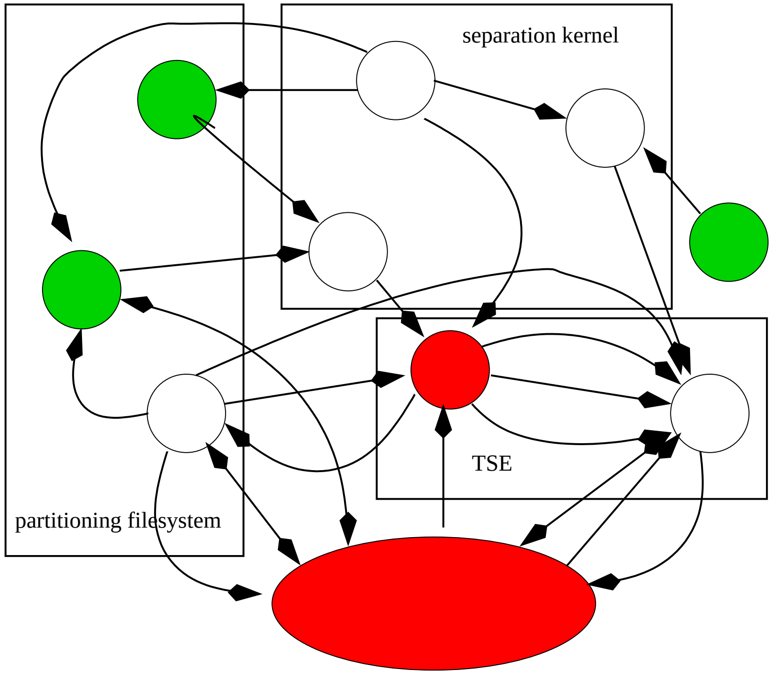
- The structure of the system implementation should directly reflect the circles and arrows picture
    - i.e., the implementation directly follows the logical design
- We can afford to have lots of circle and arrows, and should use this to reduce and simplify the trusted circles

Let me say that again

- The structure of the system implementation should directly reflect the circles and arrows picture
    - i.e., the implementation directly follows the logical design
- We can afford to have lots of circle and arrows, and should use this to reduce and simplify the trusted circles

The MILS Technology

- We can afford to have lots of circles and arrows because we can efficiently and securely share physical resources among separate logical circles and arrows
- Care and skill needed to determine which logical components share physical resources (performance, faults)



The MILS Architecture

- The MILS architecture is a combination of the idea and the technology
- Deconstruct functions so the trusted components are as simple as possible
  - These trusted components are called operational
- Allow operational and untrusted components to share resources
  - The components that do the secure sharing (separation kernel etc.) are called foundational
- We need protection profiles for these classes of components
  - Assurance specialization goes Common Criteria (CC) to Protection Profile (PP) to Security Target (ST) to Target of Evaluation (TOE)

Advantages of the MILS Architecture

- The foundational and operational security concerns are kept separate
  - Separate kinds of components
  - Separate kinds of PPs
- Cf. traditional security kernels, where one component partitioned many kinds of resources (complex implementation), and either enforced a single operational security property (too rigid to be useful) or several (too complicated to be credible)
- MILS is feasible today because we know how to do fine grain partitioning (e.g., paravirtualization), have better hardware support, and can afford the overhead

Goals for the MILS Architecture

These include

- **Security**
  - Security includes many notions, such as confidentiality, integrity, access control, authorized flow, authorized actions, and is often required in combination with other difficult properties, such as safety
- **Functionality**
  - The system must achieve its operational purpose, which is usually about something other than security
- **Assurance**
  - Need a rational approach to evaluation and certification
- **Affordability**

Previous approaches to computer security failed on one or more, or all of these

Affordability

- A reasonable expectation is that affordability will be promoted by a COTS competitive marketplace
- So we need open standards, large market, many suppliers
- The MILS component PPs (separation kernel, partitioning communication system, console, file system, network stack) are open standards intended to promote a COTS market
- Makes sense to develop these first so that suppliers have time to develop products
- But this bottom-up initiative must be complemented by a top-down one that helps systems integrators understand how to use these components
- And how to develop an evaluation case for a system from those of its components

MILS Integration Protection Profile

- Security is a system property
- Existing MILS protection profiles are for components
- How do we know that a system composed of evaluated components is secure?
    - And how is the evaluation for the system constructed from the evaluations of its components?
- This is what the MILS Integration PP (MIPP) is about
- It is an instance of compositional certification
- A bold vision that pushes the state of the art

Compositional Certification

- Because safety, security, etc. are system properties, traditional certification regimes consider only complete systems (or major components)
  - E.g., the FAA certifies only airplanes, engines, propellers
- Even when component already evaluated as part of another system, certifiers reserve right to look inside (cf. RSC)
- But modern business practices (outsourcing, COTS) make this increasingly untenable, even in first use of a component
  - System integrator, let alone system certifier, may have little visibility into the component
  - They merely define its requirements
- The component should be evaluated separately
  - Evaluation is in terms of properties delivered at interfaces
- System certification is then built on these interfaces and properties, with no looking inside

Compositional Certification for MILS

- Feasibility of compositional certification depends on the architecture
- Because compositional certification is all about properties delivered at interfaces, we need
  - Known interfaces (the paths for component interaction)
    - ⋆ There must be no paths for component interaction outside the known interfaces, even in the presence of faults, or of malice in untrusted components
  - Meaningful properties
    - ⋆ Must be meaningful at interfaces
      - ⋄ So they can be evaluated locally
    - ⋆ Must be meaningful in combination
      - ⋄ So they compose to yield evaluable system properties
- MILS is an architecture that promotes these characteristics

Two Kinds of Components, Two Kinds of PPs

The foundational and operational levels of the MILS architecture have different concerns and are realized by different kinds of components having different kinds of PPs

Operational level: components that provide or enforce application-specific security functionality

- Examples: downgrading, authentication, MLS flow
- Their PPs are concerned with the specific security function that they provide

Foundational level: components that securely share physical resources among logical entities

- Examples: separation kernel, partitioning communication system, console, file system, network stack
- Their PPs are concerned with partitioning/separation/secure sharing

Two Kinds of Components, Three Kinds of Composition

We need to consider three kinds of component compositions

operational/operational: need compositionality

foundational/operational: need composability

foundational/foundational: need additivity

Consider these in turn

Compositionality

Operational components combine in a way that ensures **compositionality**

- There's some way to calculate the properties of interacting operational components from the properties of the components (with no need to look inside), e.g.:
  - Component A guarantees P if environment ensures Q
  - Component B guarantees Q if environment ensures P
  - Conclude that $A || B$ guarantees P and Q
- Assumes components interact only through explicit computational mechanisms (e.g., shared variables)

Composability

Foundational components ensure **composability** of operational components

- Properties of a collection of interacting operational components are preserved when they are placed (suitably) in the environment provided by a collection of foundational components
- Hence foundational components do not get in the way
- And the combination is itself composable
- Hence operational components cannot interfere with each other nor with the foundational ones

Additivity

Foundational components compose with each other **additively**

- e.g., **partitioning(kernel)** + **partitioning(network)** provides **partitioning(kernel + network)**
- There is an asymmetry: partitioning network stacks and file systems and so on run as clients of the partitioning kernel

Ensuring Compositionality, Composability, and Additivity

There are two aspects

Theory: developing/applying the computer science to understand and achieve these

Application: interpreting and formulating the science in a manner consistent, as far as possible, with the CC and existing PPs

Operational PPs and Compositionality

- We might like to specify required properties of operational components in terms of information flow
- Well-known that many flow properties do not compose
  - e.g., noninterference

  They don't refine, either
- And compositional flow properties are nonintuitive
  - e.g., restrictiveness
- Much of this is because flow security is not a property
  - A property is a subset of possible traces (behaviors)
  - But we cannot tell if a given trace is flow secure without knowing what other traces there might be
- In practice, we enforce flow security by requiring something stronger that is a property (e.g., unwinding)

Operational PPs and Compositionality (ctd.)

- I suspect that if operational PPs specify claims that are properties then we can use CS-style compositional reasoning
  - E.g., assume-guarantee (seen earlier)
  - There may even be end-to-end flow interpretations of these (cf. Ron van der Meyden)
  - Gets trickier when there is physical plant involved (e.g., hybrid systems, like aircraft) because there can be interaction through the plant
- Impact on PPs: **metarequirement**

Foundational PPs

- All these deliver the claim of separation/partitioning
  - No interaction among entities (circles) except through specified channels (arrows)
- But specialized to the kind of entities considered

  Processor Partitions: separation kernel (SKPP)

  Communications: partitioning communication system (PCSPP)

  Screen real estate: console subsystem (MCSPP)

  Files: file system (MFSPP)

  TCP/IP networks: protocol stack (MNSPP)

  etc.

Foundational PPs and Composability

- This is what separation (as in separation kernel) is about
- Separation must have as its essence the guarantee of composability for operational components
- This follows (I think) when the foundational components guarantee the integrity of the interfaces of their clients
- It's not yet clear to me whether this constrains the operational PPs and their claims to have a certain form

Foundational PPs and Additivity

- I suspect we can get additivity of foundational components if all their PPs subscribe to a common notion of separation/partitioning
- And a common security environment
  - Common set of foundational threats (Mark Vanfleet)
  - Common assumptions and organizational policies
- But respecting (all and only) the essential differences among the different components
  - e.g., computation vs. storage vs. communication
- Impact on PPs: **harmonization**

The Essence of Certification

- All certification is based on **arguments** that purport to justify certain **claims**, based on documented **evidence**
- In some regimes (e.g., security), deployment decisions (i.e., judgments about the value of the claims) are separate from judgment of the credibility of the claims; in others (e.g., civil aircraft) they are combined
- Evidence may be measured facts about a system (e.g., static analysis, tests, peer review, operational history of similar systems), or claims about subsystems (supported by lower level certification)
- The evidence–arguments–claims structure is called an assurance case
- Two approaches to certification: implicit (standards based), and explicit

Assurance Cases, The CC and PPs

- The CC predated the emergence of explicit assurance cases
- But has many of their characteristics
  - Tells you what to think about, not what to do
- PPs specialize the CC requirements toward specific classes of system and component so that the developers of STs and specific TOEs have clear guidelines to follow
- Each PP should provide the framework for an explicit assurance case
  - Provides the claims
  - Specifies evidence to produce (and methods to follow)
  - Provides the argument linking evidence to claims

  It becomes a complete assurance case when the TOE developer supplies the evidence
- Need to engage with CC process to ensure consistency

Conclusions

- The MIPP is developing an approach to integrating multiple MILS components into a certifiable system
- A practical, effective, and community-endorsed approach is a prerequisite for MILS success
- So the MIPP needs to be a community effort
    - Progress is reported at quarterly meetings of the Open Group Real Time and Embedded Systems forum
    - Please attend: the MIPP needs to be a community effort
- Of necessity, the MIPP is pioneering a form of compositional certification
- Many areas want compositional certification, so MILS success would have large impact
- But wait, there's more. . .

More Conclusions

- There is general unease with software and systems (un)dependability, and emerging consensus on an approach (NRC Report, Daniel Jackson)
- There are massive recent advances in the power of automated verification and beginnings of industrial takeup (VSI, Shankar)
- There is disquiet at the costs of standards-based certification and doubts about its efficacy under modern business and development practices (COTS, outsourcing, reuse, composition, runtime adaptation)
- So it's time to develop a Science Of Certification
  - See my three papers of 2007 (just Google my name)
  - (Oh, and for MILS, see those of 1981 and 1983!)
- But wait, there's still more...

Thanks

- To those sponsoring and guiding this effort
  - Jahn Luke and Tod Reinhart of AFRL
  - Carolyn Boettcher of Raytheon
- OG RTES stalwarts
  - Joe Bergmann (OG)
  - Ben Calloni (Lockheed Martin)
  - Michael McEvilley (Mitre)
- MILS supporters and PP developers
  - SKPP folks (mostly NPS, U Idaho), Mark Vanfleet (NSA)
- My MIPP coauthor and developer of the closely related Common Criteria Authoring Environment (CCAE)
  - Rance DeLong (LynuxWorks)